**专题一：移动视觉搜索实验报告**

崔震崴 00904702 李明阳00948703 廖小泉00948166

**目 录**

[一、小组成员和分工 2](#_Toc359616594)

[二、实验内容 2](#_Toc359616595)

[三、实验设计 3](#_Toc359616596)

[1. 算法流程图 3](#_Toc359616597)

[2. 程序流程图 4](#_Toc359616598)

[3. 基本原理 4](#_Toc359616599)

[4. 程序中应用和定义的关键函数 5](#_Toc359616600)

[5. Qt界面 7](#_Toc359616601)

[四、系统预览 9](#_Toc359616602)

[五、实验分析 15](#_Toc359616603)

[1. 程序使用说明 15](#_Toc359616604)

[2. 输入文件说明 15](#_Toc359616605)

[3. 输出日志 16](#_Toc359616606)

[4. 运行环境 16](#_Toc359616607)

[5. 实验过程 17](#_Toc359616608)

[6. SIFT参数测试分析 17](#_Toc359616609)

[7. 数据集测试结果分析 21](#_Toc359616610)

[8. 遇到的问题和不足 24](#_Toc359616611)

[六、感想和体会 25](#_Toc359616612)

[七、参考文献 26](#_Toc359616613)

## 一、小组成员和分工

|  |  |  |
| --- | --- | --- |
| **姓名** | **学号** | **负责部分** |
| 崔震崴 | 00904702 | 负责核心SIFT算法的源代码调研，并使用SIFT算法来匹配图片 |
| 李明阳 | 00948703 | 使用Qt调用MobileVisualSearch.exe来实现系统的框架和外观设计 |
| 廖小泉 | 00948166 | 实现将程序用于数据集测试，就SIFT的参数进行实验，分析数据集测试、制作PPT和撰写主体报告 |

## 二、实验内容

给定几类图像数据集，每类数据集包括一种典型类型图像数据（例如：地标、物体、CD封面、图书封面、视频帧等）。

我们小组编写一个视觉搜索程序，主体功能模块包括图像特征提取以及基于KDTree的匹配查找。给定一张查询图像，能够返回其在上述数据集的相关图像搜索结果，搜索精度越高越好。

一个完整的视觉搜索系统主要包括图像局部特征提取模块、基于词汇包的倒排索引模块（未使用，改为KDTree最近邻查找匹配）、PC客户端开发。上述三个模块由我们三位同学分工协作完成。具体实验基本内容为：图像局部不变特征的提取；利用SIFT或其他局部不变性特征，开展图像匹配实验，验证特征的有效性。

实验目的是熟悉表征图像内容的视觉特征，并利用视觉特征进行图像匹配与检索。

## 三、实验设计

### 1. 算法流程图



图1 算法流程图

### 2. 程序流程图



图2 程序流程图

### 3. 基本原理

**KeyPoint：**是用于图像特征点描述。

**Descriptor：**描述子，本程序应用的是SIFT算法及其对应的Features Descriptors。

**DescriptorMatcher：**特征描述子的匹配类，主要采用了FlannBased、BruteForce、BruteForce-L1三种，其中FlannBased时间效率最高，默认采用。

**准确度计算：**在一个图片集中，准确度=成功匹配查询图片的数目/总的匹配数目\*100%，总的匹配数目=查询图片的数目\*每张查询图片要求匹配图片的数目。准确度越高，说明该匹配算法越好。

**时间记录：**记录每张图片高斯平滑、提取Keypoints和生产FeaturesDescriptors的时间，还有训练集建立KDtree的时间，每张查询图片匹配的时间。

**SIFT:** SIFT特征（Scale-invariant feature transform，尺度不变特征转换)是一种电脑视觉的算法用来侦测与描述影像中的局部性特征，它在空间尺度中寻找极值点，并提取出其位置、尺度、旋转不变量。其应用范围包含物体辨识、机器人地图感知与导航、影像缝合、3D模型建立、手势辨识、影像追踪和动作比对。

**KDTree：**KDTree是做多维数据索引时候用到的一种数据结构，KDTree是二叉检索树的扩展，KDTree的每一层将空间分成多个，树的顶层结点按一维进行划分，下一层结点按另一维进行划分，以此类推，各个维循环往复。划分要使得在每个结点，大约一半存储在子树中的点落入一侧，而另一半落入另一侧。当一个结点中的点数少于给定的最大点数时，划分结束。用KDTree有助于提升匹配的效率。

### 4. 程序中应用和定义的关键函数

**class siftmy**

/\*sift的opencv版本，主要用来研究其算法和更改相关参数\*/

**siftmy::unpackOctavebool readImgNames(const string &setName, const int resImgNum)**

/\*用来确定金字塔的层数\*/

**void siftmy::operator()**

/\*通过重载，实现sift的主要步骤\*/

**bool readImgNames(const string &setName, const int resImgNum)**

/\*自己定义读取图片集setName中的图片名字，resImgNum表示读取图片的数目\*/

**bool trainSet(Ptr<DescriptorMatcher>& descriptorMatcher)**

/\*自己定义用来建立训练集的描述子匹配器集的函数，descriptorMatcher 表示描述子匹配器集\*/

**bool matchImgs(const string &queryImgName, const int resImgNum, vector<string> &resultImgNames, Ptr<DescriptorMatcher>& descriptorMatcher)**

/\*查询图片用来匹配图片的函数，

\*queryImgName 表示查询图片的路径名称

\*resImgNum 表示每张图片匹配的数目

\*resultImgNames 表示存储匹配结果图片的名字容器集

\*descriptorMatcher 表示描述子匹配器集\*/

**void DescriptorMatcher::train()**

/\*训练集的特征描述子进行训练，建立匹配器\*/

**void DescriptorMatcher::knnMatch(const Mat& queryDescriptors, vector<vector<DMatch>>& matches, int k, const vector<Mat>& masks=vector<Mat>(),bool compactResult=false )**

/\*queryDescriptors –Query set of descriptors.

masks –Set of masks. Each masks[i] specifies permissible matches between the input query descriptors and stored train descriptors from the i-th image trainDescCollection[i].

matches –Matches. Each matches[i] is k or less matches for the same query descriptor.

k –Count of best matches found per each query descriptor or less if a query descriptor has less than k possible matches in total.

compactResult –Parameter used when the mask (or masks) is not empty. If compactResult is false, the matches vector has the same size as queryDescriptors rows. If compactResult is true, the matches vector does not contain matches for fully masked-out query descriptors.

These extended variants of DescriptorMatcher::match() methods find several best matches for each query descriptor.

The matches are returned in the distance increasing order.\*/

**Ptr<DescriptorMatcher> descriptorMatcher = DescriptorMatcher::create(const string& descriptorMatcherType)**

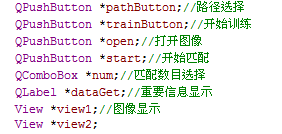
/\*特征描述子匹配的算法：descriptorMatcherType采用FlannBased算法时间效率最高，还可以采用BruteForce, BruteForce-L1 \*/

### 5. Qt界面

使用Qt5进行编程，虽然Qt也是基于c++的库，但是由于所用库不同，在一个工程中直接编写较为繁琐，使用调用可执行文件的方法来完成调用关系。

界面包括：

* 两个图片展示停靠部件，用于展示所选择图像和匹配图像。
* “训练集路径”选择按钮、“开始训练”按钮、“打开图像”按钮、匹配数量选择下拉菜单、“开始匹配”按钮
* 文本显示框，用于提示相应信息。
* 文件及路径选择窗口
* 下图为主要的部件定义



界面另外一个重要的部分是和可执行文件的交互。Qt编程中需要新建一个进程，调用相应的exe文件。在进程交互的过程中，由于是本机通讯，没有使用socket，而是直接使用两个进程的输入流和输出流进行交互。不过需要注意的是，在读取数据时要将不必要的换行符等信息去除，否则将会出错。

## 四、系统预览

程序运行的进入界面（图3）：

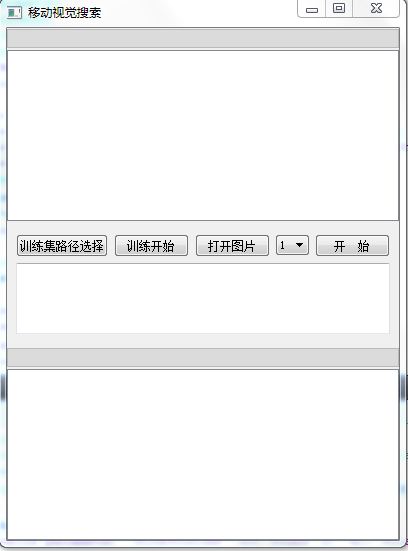


图3

首先，点击“训练集路径选择”选择训练集，点击“训练开始”按钮，将进行训练集图片的特征提取、建立，形成KDtree（图4）：

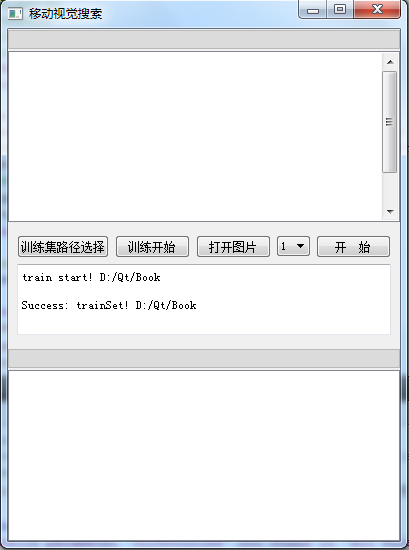


图4

接着，点击“打开图片”按钮，选择一张待查询的图片，并选择该图片匹配返回的图片数目（可以选择的范围：1-5）（图5）：

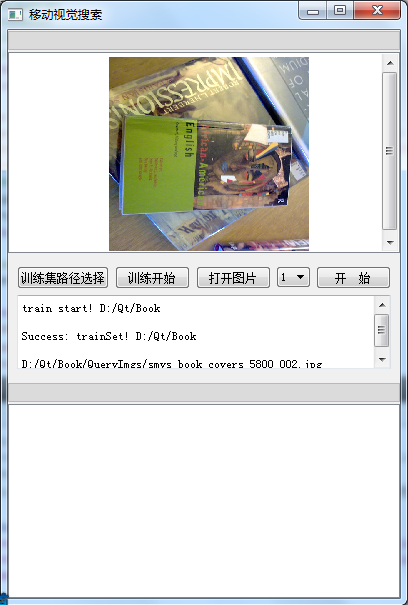


图5

之后，点击“开始”按钮，进行图片的匹配过程，返回匹配的结果（图6）：

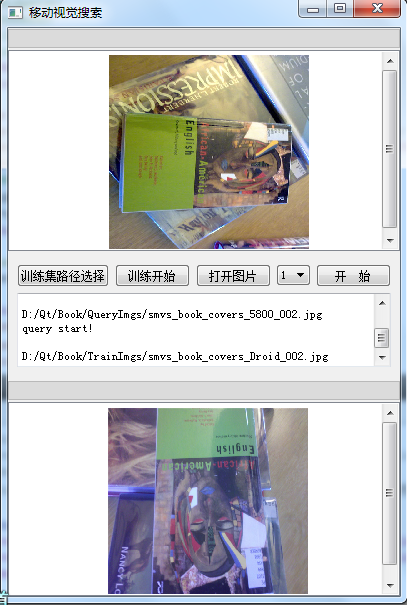


图6

可以选择匹配图片返回2张、3张，点击“开始”按钮继续匹配，结果如图7（匹配2张）、图8（匹配3张）：



图7

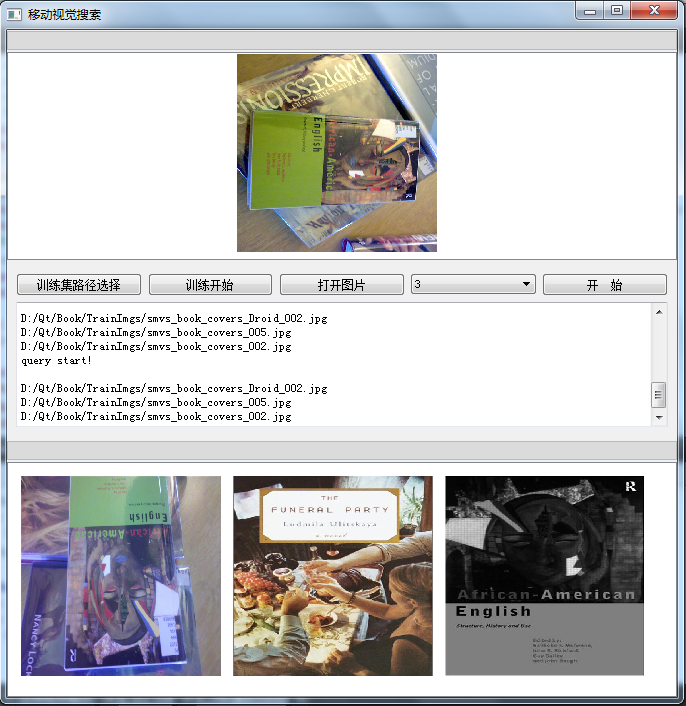


图8

## 五、实验分析

### 1. 程序使用说明

如果使用Qt Creator，打开Qt\_code文件夹内的工程文件的gui（类型是Qt Project File），注意要把exe文件的路径D:/Qt/MobileVisualSearch.exe（可以在文档mainwindow.cpp内修改路径），才能正常运行。

如果不适用Qt Creator，可以使用DOS界面（或者双击直接运行MobileVisualSearch.exe文件），注意输入按照如下要求：

>>train //输入train表示首先对训练集图片进行训练

>>D:/Qt/Book //输入训练集的路径（用本文件夹内的Book数据集，已经按照要求处理过）

待出现Success……之后；

>>query //输入query表示对输入查询图片进行匹配

>>D://Qt/Book/QueryImgs/smvs\_book\_covers\_5800\_001.jpg //输入查询图片的绝对路径

>>2 //输入2表示每张图片需要匹配返回的数目，可以为1、2、3、4……

待返回匹配结果，以图片的路径显示，可以继续上述的训练或者查询；

>>stop //输入stop表示退出程序

### 2. 输入文件说明

**trainImgNames.txt文件：**训练集图片的名字，在Book文件夹下

Reference/smvs\_book\_covers\_001.jpg

Reference/smvs\_book\_covers\_002.jpg

……

为了便于展示和快速得到结果，我们对Book数据集进行了处理，一个是因为内存限制，我们将大于1MB的图片进行压缩处理，使其分辨率缩小为原来的1/10大小。

将Book的训练集TrainImgs的图片有200张减小为10张，仅仅是因为展示的原因，可以支持快速演示，程序支持几百张乃至千张数量级的图片，不过随着数据集的扩大，匹配的时间会增大。

### 3. 输出日志

输出的文件为log.txt（在exe程序所在的文件目录下）

内容包含程序运行过程中的各类信息，数据用于后期的准确度和效率分析。

### 4. 运行环境

(1) 电脑系统配置

电脑型号：惠普 HP ProBook 4411s 笔记本电脑

操作系统：Windows 7 专业版 32位 SP1 ( DirectX 11 )

处理器：英特尔 酷睿2 双核 T6570 @ 2.10GHz 笔记本处理器

主板：惠普 3074 (英特尔 4 Series - ICH9M 笔记本芯片组)

内存：2 GB ( 威刚 DDR2 800MHz )

主硬盘：西数 WDC WD2500BEKT-60F3T1 ( 250 GB / 7200 转/分 )

显卡：ATI Mobility Radeon HD 4330 ( 512 MB / 惠普 )

(2) Visual Studio

Microsoft Visual Studio Ultimate2012：版本11.0.60315.01 Update2

Microsoft .NET Framework：版本4.5.50709

Microsoft Visual C++ 2012

(3) OpenCV 2.4.4

(4) Qt Creator 2.7.0基于 Qt 5.0.2 (32 bit)

### 5. 实验过程

1. 单图测试：主要是对单个图像的处理。首先，对比不同sift参数下图像特征点的分布，来判断参数的范围。然后我们又做了BOW聚类后用knnmatch匹配。得到的结果是参数基本上都在默认值附近，BOW+knnmatch的方法得到的匹配效果很差，如图9：

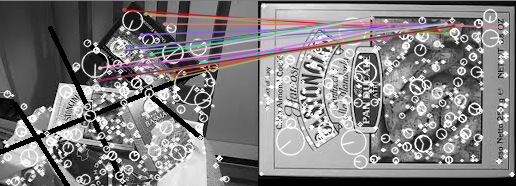


图9

1. 多图测试：我们用100张图像为数据集，对比匹配时间和匹配度，发现最好的情况下是在默认参数左右的情况下。（贴对比的结果）

在这一步我们还尝试做了优化，具体方法为用匹配过程中得到的距离取其最大值max和最小值min，利用（max-min）/max与阈值t对比，当比t小时则这张图片不作为匹配的图片。

1. 实现程序：利用我们在前两部的结论完成最终程序。

### 6. SIFT参数测试分析

（1） 参数SIFT\_SIGMA：default sigma for initial gaussian smoothing

表1 参数SIFT\_SIGMA对图片查询的影响实验结果

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| SIFT\_SIGMA | pyramid | keypoint | descriptor | 匹配子提取总时间 | keyPoint\_num | success\_good\_matches |
| 0.8 | 504.38 | 586.77 | 99.72 | 1190.87 | 206 | 0 |
| 1.0 | 580.48 | 601.38 | 152.09 | 1333.95 | 185 | 1 |
| 1.2 | 677.65 | 605.54 | 199.73 | 1482.91 | 187 | 1 |
| 1.4 | 763.44 | 614.68 | 260.76 | 1638.88 | 189 | 1 |
| **1.6** | **836.60** | **631.79** | **368.75** | **1837.14** | **178** | **1** |
| 1.8 | 895.68 | 670.87 | 453.01 | 2019.56 | 189 | 1 |
| 2.0 | 989.99 | 692.64 | 527.72 | 2210.35 | 177 | 1 |
| 2.4 | 1142.02 | 708.69 | 707.21 | 2557.92 | 178 | 1 |

图10 变量SIFT\_SIGMA实验结果

从表1和图10的折线图中可以看出，SIFT\_SIGMA在从0.8增大到2.4的过程中，提取图片的匹配子的时间逐渐增大，而keyPoint的数量却呈下降趋势。

（2）参数SIFT\_CONTR\_THR：default threshold on keypoint contrast |D(x)|

表2 参数SIFT\_CONTR\_THR对图片查询的影响实验结果

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| SIFT\_CONTR\_THR | pyramid | keypoint | descriptor | 匹配子提取总时间 | keyPoint\_num | success\_good\_matches |
| 0.01 | 834.47 | 850.75 | 833.01 | 2518.23 | 452 | 0 |
| 0.02 | 814.55 | 776.82 | 542.60 | 2133.97 | 273 | 1 |
| 0.03 | 829.08 | 645.40 | 428.96 | 1903.44 | 215 | 1 |
| **0.04** | **821.64** | **628.77** | **337.23** | **1787.64** | **178** | **1** |
| 0.05 | 819.22 | 605.24 | 304.64 | 1729.10 | 152 | 1 |
| 0.06 | 817.46 | 606.85 | 269.40 | 1693.70 | 133 | 1 |
| 0.07 | 819.00 | 586.68 | 209.72 | 1615.39 | 103 | 1 |

图11 变量SIFT\_CONTR\_THR实验结果

从表2和图11的折线中可以看出，参数SIFT\_CONTR\_THR在从0.01增大到0.07的过程中，keyPoint的数目在逐渐减少，且图片提取匹配子的时间也在下降。

（3）参数nOctaveLayers

表3 参数nOctaveLayers对图片查询的影响实验结果

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| nOctaveLayers | pyramid | keypoint | descriptor | 匹配子提取总时间 | keyPoint\_num | success\_good\_matches |
| 2 | 877.67 | 481.26 | 376.23 | 1735.16 | 174 | 0 |
| **3** | **826.55** | **630.33** | **335.04** | **1791.92** | **175** | **1** |
| 4 | 827.79 | 861.85 | 302.36 | 1992.00 | 176 | 1 |
| 5 | 909.02 | 990.25 | 306.64 | 2205.91 | 187 | 1 |
| 6 | 969.34 | 1418.89 | 321.70 | 2709.93 | 199 | 1 |

图12 变量nOctaveLayers实验结果

从表3和图12的折线中可以看出参数nOctaveLayers在从2增加到6的过程中，keyPoint的数目在逐渐增加，且图片提取匹配子的时间也在增加。

我们小组一共进行了7个参数的实验，其中只有以上这3个有较为明显的效果，故将其提出来，其他的结果可以参见Excel表格：参数实验.xlsx。

### 7. 数据集测试结果分析

我们采用了不同的方式，主要是DescriptorMatcher参数（FlannBased、）和VS的调试模式：Debug和Release。

表4 集中匹配法的时间对比、匹配结果数目

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| DescriptorMatcher | train\_time | match\_time | query\_total\_time | match\_num | success | fail | percent |
| BruteForce-L1 | 6587.78 | 80133.39 | 83068.65 | 600 | 499 | 101 | 83.17% |
| BruteForce | 6254.01 | 79863.54 | 82713.95 | 600 | 504 | 96 | 84.00% |
| FlannBased | 6483.57 | 5534.91 | 8486.08 | 600 | 497 | 103 | 82.83% |
| BruteForce-Release | 1060.51 | 21924.39 | 22411.85 | 600 | 504 | 96 | 84.00% |
| FlannBased-Release | 1050.54 | 107.98 | 544.37 | 600 | 491 | 109 | 81.83% |

图13 几种匹配法的时间对比

图14 几种匹配法的匹配结果数目

从表4的数据中可以看出，几种匹配法的匹配结果都还可以，达到80%左右，相差不大。但从时间上来看，FlannBased匹配法由于BruteForce和BruteForce-L1，Release比Debug速度更快。

表5 不同数据集的匹配时间、匹配数目

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| PictureSet: FlannBased | train\_time | match\_time | query\_total\_time | match\_num | success | fail | percent |
| Card | 6483.57 | 5534.91 | 8486.08 | 600 | 497 | 103 | 82.83% |
| Book | 3126.72 | 7038.01 | 11830.50 | 600 | 491 | 109 | 81.83% |
| CD | 5131.68 | 6333.23 | 10008.39 | 600 | 472 | 128 | 78.67% |

图15 几个数据集单张图片匹配的平均时间

图16 几个数据集的匹配结果数目

图17 单张图片查询各阶段花费时间

从表5和图15、图16、图17可以看出，几个数据集的匹配度在80%左右，在匹配时间上，Card数据集快于CD数据集，CDs数据集快于Book数据集，这根它们的图片性质（特征点的数目）有关。

### 8. 遇到的问题和不足

目前系统还比较不完善，没有采用BOW和SVM训练的方式，直接使用了OpenCV自带的knnMatch()（使用了KDTree和最近邻匹配算法）来匹配图片。但是从数据集的测试结果来看，时间效率和匹配度都还可以。

之所以没有开发移动客户端，主要也是一种畏难心理，在参考往年同学实现客户端的Project之后，发现需要比较复杂和麻烦的编程实现，鉴于能力和精力有限，就没有实现。

## 六、感想和体会

本次实验，相对我们小组三人来说，是一个挑战，在之前的课堂上，我们从老师的关于移动视觉搜索的应用讲解中，大致了解了这个实验所要实现的功能，这个实验非常具有实用性和社会效用，因此我们小组决定选择这个实验来编程实现。

前期，我们小组对实验涉及到的论文、书籍和OpenCV 的具体操作、函数实现进行了阅读和总结，认为OpenCV 在不断更新版本的同时提供了许许多多非常有效的函数，基于前人的基础上，我们大量使用了OpenCV 2.4.4 自带的许多库函数，节省了自己进行匹配的编程实现，较为概括地用总体流程进行了整个实验的流程。

中期，具体编程把各个函数使用嵌入到自己的程序流程中，利用文件输入输出进行对程序运行进行了记录，用于后期的数据分析。特别是受制于电脑内存，对于较高分辨率的图片，往往会造成内存溢出，无法运行，在老师的指点下，缩小了图片的分辨率和减少了图片提取特征的数目，但是对匹配效果影响甚小。

后期，根据数据文件进行了数据分析，提取了图片高斯平滑时间、Keypoints获取时间和FeatureDescriptors建立时间、建立KDtree总时间、匹配成功数和错误数、每张图提取的特征数目和匹配时间，基于以上数据计算出每张图片的平均时间、匹配度等。根据表格的数据，用条形图和折线图进行对比分析，给出基本的结论。

在课堂展示环节，我们进行了整体流程的介绍，并做了简单的程序演示。老师针对小组存在的一些问题，给出了建议，之后我们小组对建议进行了分析和讨论，采纳了适合我们小组和便于实现的建议，对实验进行了完善。

经过整个实验，我们三人对于视觉移动搜索的原理和方法均有较为深入的理解，对OpenCV的使用有了初步的掌握和经验，在今后的学习过程中，我们将这些作为宝贵的财富加以珍惜和实践。

## 七、参考文献

[1] [【OpenCV】SIFT原理与源码分析](http://blog.csdn.net/xiaowei_cqu/article/details/8069548)，http://blog.csdn.net/xiaowei\_cqu/article/details/8069548

[2] [【OpenCV】特征检测器 FeatureDetector](http://blog.csdn.net/xiaowei_cqu/article/details/8652096)， <http://blog.csdn.net/xiaowei_cqu/article/details/8652096>

[3] 王永明，王贵锦编著，《图像局部不变性特征与描述》，北京：国防工业出版社，2010年